

Concerns:

Many play-back programs today don't even have a zoom option. To this extent, mode "b" may not be needed.

Mode c could be implemented today with 16bit/pel RGB (555) in s/w and the only improvement we provide is 24bit/pel capability and h/w color conversion (some level of acceleration). So mode "c" may not be needed as a special h/w.

If this is the case, maybe we do not need to do anything in h/w for Cinepak !

One idea: should we run 24bit/pel MS windows stored in 4:2:2 (Y0 U01 Y1 U01) format? This will allow us to run 24bit/pel MS Windows and use 16bit/pel memory space with no loss of quality. The windows driver could compress the data and the h/w could decompress it.

4.4 Nordic-1M Motion Video Architecture

Sasha Eglit, Rakesh Bindlish, Vlad Bril and Dave Keene are important contributors to the Motion Video Architecture definition.

4.4.0 The Problems to Solve & Generic Solutions

Nordic-1M is supposed to support CDROM play-back and live-video under Microsoft's Video for Windows. With all of today's VGA Compatible Graphics Controllers, the play-back picture pixel depth has to be the same as the surrounding MS Windows pixel depth. In other words, we have to run MS Windows in a 24bit/pel mode in order to display 24bit/pel in the play-back window.

The only way we can run today live video is from a Video Port (= Feature Connector). In this case we'll use an overlay or a color key to define the live video window.

Due to CDROM player, s/w interface complexity, CPU bus and Video Memory limitations, only small clips at 15fps or less can be playd-back today. So today we need a lot of Video Memory to run small clips in slow motion at low rezolution. Nordic-1M will try to use less Video Memory, increase the clips size, their resolution and speed.

Cirrus Confidential
Business Information

Nordic-1M is supposed to support:

- accelerated play-back for all popular CDROM compression standards, especially Cinepak and Indeo
- live video from a PCMCIA card in a system with a standard PCMCIA host adapter, for a "Multimedia-Ready Notebook Computer"
- live video from a Feature Connector, preferably VESA Advanced Feature Connector compatible, for a Multimedia Notebook Computer with direct NTSC input (full mother-

board solution).

What will Nordic-1M bring to the Video plate:

- A. Nordic-1M will break the dependency between the pixel depth of MS Windows and the pixel depth of the live-video / play-back window. By doing so, Nordic-1M will be able to run 24bit/pel LV/PB while running MS Windows in 8bit/pel or 4bit/pel (even planar). This will lead to high quality live-video and play-back, while minimizing Video Memory requirements.
- B. Nordic-1M will further reduce Video Memory requirements as well as Video Memory Bandwidth requirements by storing data in compressed form: 4:2:2 YUV (16bit/pel), 4:1:1 YUV (12bit/pel) or Sashapak YUV (2.6bit/pel)
- C. Nordic-1M will define one architecture and work-frame to run both play-back and live-video in a unitary way for both s/w and h/w.
- D. Nordic-1M will support up to two active LV/PB windows at the same time.
- E. By providing h/w YUV->RGB conversion and 1:2 zoom Nordic-1M will accelerate playback for most standards, but mostly for standards that allow us access to their decompressor like Cinepak (for whose decompressor we have a licence).

Nordic-1M will not provide Cinepak specific acceleration (custom BLT).

On the Live Video from Feature Connector (or Video Port)

Due to pin-out limitations, the VAFC implementation has to be limited to 8 pins of data requiring external muxing of 16bit data to Nordic. Further, the VAFC solution will be restricted to 5428 type support: overlay and color key (no memory video port). The only additions to the 5428 support will be the internal YUV-> RGB converter able to accept sequential 4:2:2 YUV (16bit/pel in average: Y0,U,Y1,V) and display it as a 24bit/pel RGB in the overlay/color-key window. We may also support horizontal 1:2 zoom with averaging. Nordic-1M will be back-wards compatible to 5428 and it will support also 16bit 5-5-5 RGB Sierra and 5-6-5 RGB XGA pixel formats. As in 5428, live video from the Feature Connector can be executed only from mode 5F (640x480 256 colors).

In the following we will refer mostly to the other two modes of operation: play-back and live video from a PCMCIA card, which we'll call compressed live video, because due to PCMCIA bus bandwidth limitations, we assume that video data received by Nordic is compressed with Sashapak (see Sashapak details further in this spec). Please note that due to its huge advantages, it makes sense to use Sashapak as a mother-board solution too. The main reason we continue to support the Feature Connector based live video is the availability of drivers for 5428 which will give Nordic an early start in the live video arena

Cirrus Confidential
Business Information

CL 99792

Video Memory Band-width Constraints

Please note that, even with a 32bit wide Video Memory, there are severe memory bandwidth limitations when running 16bit/pel and 24bit/pel graphics modes.

Assuming that Nordic-1M does 8 CRT fetches in a row and 1 CPU cycle, for a CRT or TFT panel, here are the minimum frequency requirements for memory clock frequency at different resolutions and pixel depths:

- 24bit/pel:

$R+7P+R=12m+14m=26m$ with 1 CPU cycle per CRT FIFO fill fetches data for 32B/3B=10pixels

$R+7P = 6m+14m = 20m$ with no CPU cycle during non-display time

min mclk freq = 65MHz / 50MHz @ 640x480 -> Nordic-1M upper limit at 5V CVDD

= 104MHz / 80MHz @ 800x600 -> too high

= 169MHz / 133MHz @ 1024x768 60Hz -> too high

16bit/pel:

$R+7P+R=12m+14m=26m$ / $R+7P = 20m$ fetches data for 32B/2B=16pixels

min mclk freq = 40.6MHz / 36MHz @ 640x480 -> OK even at 3V

= 65MHz / 50MHz @ 800x600 -> Nordic-1M upper limit at 5V CVDD

= 105MHz / 83MHz @ 1024x768 60Hz -> too high

12bit/pel:

$R+7P+R=12m+14m=26m$ / $R+7P=20m$ fetches data for 32B/1.5B=21pixels

min mclk freq = 30.9MHz / 23MHz @ 640x480 -> OK even at 3V

= 49.5MHz / 38MHz @ 800x600 -> OK even at 3V

= 80.4MHz / 64MHz @ 1024x768 60Hz -> too high

8bit/pel:

$R+7P+R=12m+14m=26m$ fetches data for 32B/1B=32pixels

min mclk freq = 20.3MHz @ 640x480 -> OK even at 3V

= 40.6MHz @ 800x600 -> OK even at 3V

= 52.8MHz @ 1024x768 60Hz -> OK at 5V

We may conclude that Nordic-1M will be able to run 24bit/pel only at 640x480 resolution. 16bit/pel up to 800x600. 12bit/pel up to 800x600 and 8bit/pel up to 1024x768 resolution. These results apply only to CRT or TFT and single scan STN panels. Please note that for dual scan STN panels the memory requirements increase significantly.

These severe memory bandwidth limitations lead us to attempt to minimize video memory traffic. If we store Cinepak data in 4:2:2, with an average of 16bit/pel, independent of Cinepak window size, Nordic-1M will not be able to run Cinepak above 800x600. Any attempt to execute vertical zoom with interpolation, even with two points interpolation will further aggravate this memory band-width bottleneck. The next step should be 100MHz SDRAM-s with a large CRT FIFO (32x32) or 64bit wide memory data bus. These considerations apply to any playback compression standard that stores data in video-memory as 16bit/pel in average. The key to a successful compression standard would be one that does decompression on the CRT memory read and stores data in memory as 8bit/pel or less, in average.

As we store live video at 2.6bit/pel and fetch it as if it was 8bit/pel with Sashapak, Nordic-1M should be able to run live video even at 1024x768.

If we defined a new CDROM play-back standard based on Sashapak, then we could run even 1024x768 with 24bit/pel accuracy and keep it memory as an 8bit/pel. Keith and Ken are actually working on it, with Sasha's help.

4.4.1 On Sashapak

Cirrus Confidential
Business Information

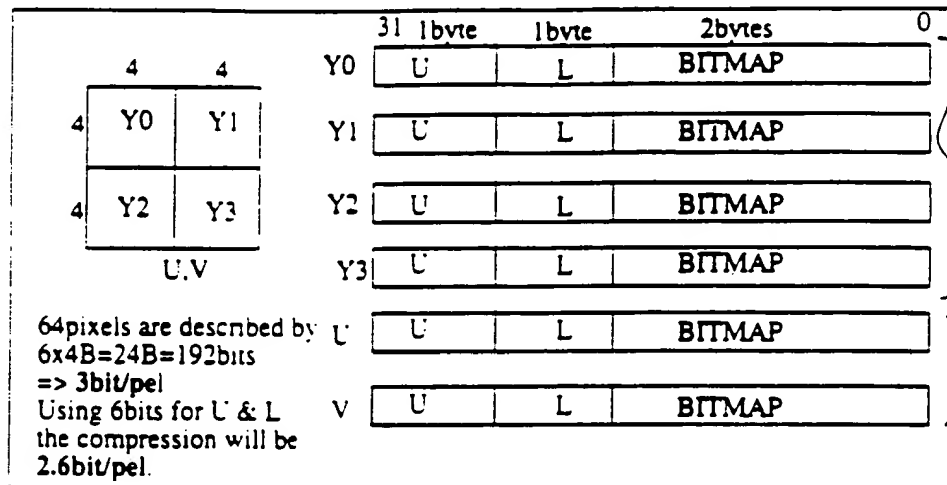
CL 99793

Cirrus Logic, Inc. Confidential Information

Sashapak is a highly asymmetric, lossy "Block Truncation" compression algorithm optimized for visually equivalent image processing.

Data is processed as Y,U,V one byte each. U and V are sub-sampled, characterizing one 8 by 8 pixel array, while Y is characterizing a 4x4 pixel array. Actually one of two values is assigned to each pixel inside the 4x4 (for Y) and 8x8 (for U and V).

At compression time, the compression chip determines a threshold and two values U&L for each group of sixteen (4x4) pixels. This way all pixels above threshold will be assigned the value U, all pixels below threshold will be assigned the value L.



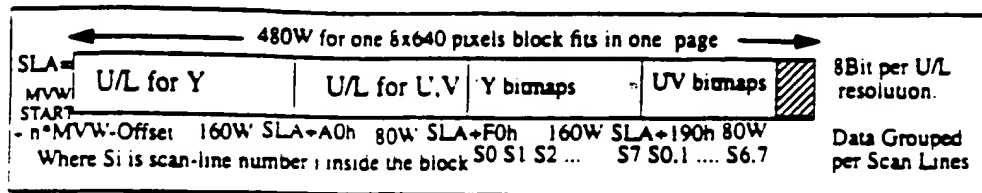
At decompression, the bitmap will control a mux between the U and L values. This way the decompression is straightforward.

The main drawback of this approach is that a BITMAP contains Y data from 4 scan lines and U,V data from 8 scan lines. When data is read from memory to be displayed only data from one scan line is used. Similarly the U/L values apply to several scan-lines and the same value will be read in each scan line. The same data will be read again and again in each scan line raising the actual memory bandwidth requirements to an equivalent of 16bit/pel map.

In order to overcome this problem and reduce memory bandwidth requirements, Sasha proposed a scan line oriented, segregated mode of data organization. This would require the Sashapak compression chip to write compressed data in a specific way putting the strain on the compression chip address generator. The basic principles of the new organizations are:

- 1 Data is organised in the MV Window in chunks of 32 sequential pixels.
 - a) For 8bit U/L resolution, as one Y covers 4 pixels, 8 U/L sequential values are needed, 16bits each -> $8 \times 16 / 32 = 4W$ ($8 \times 12 / 32 = 3W$ for 6bit res. U/L) (where W is a 32bit word).
 - b) U and V U/L data is packed together in the same word (16bits for U U/L and 16 bits for V U/L -> total 32bits=1W) and as one pair of U and V applies to 8pixels, 4 such values are needed for 32 sequential pixels -> $4 \times 32 = 4W$. ($4 \times 12 \times 2 / 32 = 3W$ for 6bit res. U/L)
 - c) The mask data is also scan-line oriented: all 32bits of Y mask of sequential 32pixels are placed in one 32bit word and due to sub-sampling on U and V, all 32bits of U and V mask of sequential 32pixels are placed in one 32bit word. So in 2W data for 32 sequential pixels on one scan line is packed. -> 2W (the same for 6bit resolution for U/L)

d) For 640x480 window size, even at 8bit U/L resolution, data for 8 scan-lines fits in one DRAM page (symetric DRAM-s assumed): $320WY+80WU+80WV=480W < 512W$ page



f) The key to memory bandwidth optimization is the fact the the compression chip will organize data sequentially over 32 pixels instead of 8. This will be done for the mask bits which will be organized sequentially by scan line and for the U/L data. This way a 32bit word of Mask data contains only information for the current scan line and no extra memory fetches are required. The main reason we segregate U/L for Y and U/V is the fact that we want to use 6bit U/L. In this case it is easier to identify the proper U/L info chained within 32bit words if the words are sequential and we have a fix relationship between their address and the 6bit U/L quantities.

- g) If we calculate now the amount of data we need to fetch to display 32 sequential pixels:
- for 8bit per U/L: $4W (YU/L) + 4W (UV U/L) + 1W (YM) + 1W (UVM)=10W=40B$
 $--> 40B/32pixels=320bit/32pixels=10bit/pel$ to fetch from memory
 - for 6bit per U/L: $3W (YU/L) + 3W (UV U/L) + 1W (YM) + 1W (UVM)= 8W=32B$
 $--> 32B/32pixels = 8bit/pixel$

This is a very good data fetch ratio, much better than the 16bit/pel we'd get without the optimization.

g) When reading data from the Motion Video Window the controller will generate the following sequence of addresses, all of them in the same DRAM page (8bit per U/L):

- fetch the YU/L for the first 32 pixels on scan-line n at addresses:
 $MVW_Start+n*MVW_offset, +1, +2, +3$
- fetch the UV U/L for the first 32 pixels on scan-line n at address:
 $MVW_Start+n*MVW_offset+A0h, +1, +2, +3$
- fetch the Y BitMap for the first 32 pixels on scan-line n at address:
 $MVW_Start+n*MVW_offset+F0h$
- fetch the UV BitMap for the first 32 pixels on scan-line n at address:
 $MVW_Start+n*MVW_offset+190h$
- fetch the YU/L for the first 32 pixels on scan-line n at addresses:
 $MVW_Start+n*MVW_offset+4, +5, +6, +7$
- fetch the UV U/L for the first 32 pixels on scan-line n at address:
 $MVW_Start+n*MVW_offset+A0h+4, +5, +6, +7$
- fetch the Y BitMap for the first 32 pixels on scan-line n at address:
 $MVW_Start+n*MVW_offset+F0h+1$
- fetch the UV BitMap for the first 32 pixels on scan-line n at address:
 $MVW_Start+n*MVW_offset+190h+1$
- now the MVW FIFO is full.
- chunks of ten 32bit words will be processed for every 32pixels displayed
- the MVW FIFO gets empty after the first 10 words being read for decompression and display.

Cirrus Confidential
Business Information

CL 99795

Please note that all data for every 8 scanlines starting from the top of the MVW is in the

same DRAM page.

Because we have to fetch the MVW Data before we start displaying it, and because at that time the CRT FIFO does not empty as fast as it is needed (unless MS Windows is running at the same pixel depth as the MV Window), it is necessary to have a separate FIFO for the MVW or at least half the FIFO (at least 10W for 8bit per U/L and at least 8W for 6bits per U/L).

h) When we run 6bit per U/L, because the Video Memory band-width requirements are as if we were running 8bit/pel, we could do vertical zoom with averaging. But there should not be a need for zoom with Live Video because there is enough CPU and memory band-width to run 640x480 windows even when MS Windows is run as 1024x768 4 or 8bit/pel.

Conclusion: With Sashapak we can get data organized in memory such that the memory bandwidth is as for 10bit/pel (8bit U/L) or 8bit/pel (6bit U/L) pixel depth. The actual memory area will be close to 3bit/pel or 2.6bit/pel depending on the resolution of U/L values - 8 or 6bits per value.

Please note that the memory area will be slightly larger due to the fact that we use only 480 words in a page of 512.

4.4.2 YUV to RGB Conversion Algorithm

The ideal YUV to RGB transformation is:

$$R=Y+1.37V$$

$$B=Y+1.73U$$

$$G=Y-0.699V-0.336U$$

After defining an objective way of measuring color error, Sasha came up with the following Conversion algorithm:

$$R=Y+1.375V=Y+1\frac{3}{8}V$$

$$B=Y+1.75U=Y+1\frac{3}{4}U$$

$$G=Y-0.375U-0.75V=Y-(3/8U+3/4V)$$

This algorithm can be implemented with 8 9bit adders and it should support excess 128 or two's complement U,V formats (U,V may be negative while Y is always positive).

This algorithm will be used with Live Video.

For Cinepak, the YUV->RGB converter will be reconfigured to the Cinepak YUV->RGB conversion rule:

$$R=V+Y$$

$$B=U+Y$$

$$G=Y-V/2-U/4$$

**Cirrus Confidential
Business Information**

CL 99796

4.4.2 Motion Video Architecture Functional Blocks

Instead of treating play-back and live-video as two separate modules, we'll isolate several generic functions to be used by both:

- Display Window Generation Block:

allows s/w to define one or two play-back or live video window(s).

The MV Window is defined in memory cycles relative to the start of line in horizontal and frame in vertical direction. Horizontally the unit of measure is different outside relative to the inside of the window: the horizontal window start coordinate is expressed in memory cycles for the surrounding pixel depth (MS Windows pixel depth, normally 4 or 8 bit/pel), while the width of the MVW is expressed in memory cycles for the MVW pixel depth (8bit RGB, 16bitRGB, 422YUV spacial, 422 YUV linear, 411YUV linear, Sashapak).

- **XS = MVW Horizontal Start Coordinate** (in surrounding pel depth memory cycles)

The horizontal position is programmed with 8 pixel resolution (0,8,16,... pixels starting from the left side of the screen)

XS register CR34[7:0] -> 8 bits

(up to 1K pels in increments of 8 pels)

To get an early warning, we will require XS to be programmed 8 pixels less than the actual position: 0 is zero, 1 is 16pels, 2 is 24 pels....

- **XW = MVW Horizontal Width** (MVW pel depth memory cycles)

XW register CR35 [7:0] -> 8 bits

(up to 1K pels in increments of 8 pels)

To get an early warning, we will require XW to be programmed 8 pixels less than the actual width: 0 is zero, 1 is 16pels, 2 is 24 pels.... width.

- **YS = Vertical MVW Start** (in true scan-lines not affected by scan-line doubling or panel vertical expansion)

10 bits in CR36[1:0] (msb-s) and CR37[7:0] (lsb-s).

- **YE = Vertical MVW End**

(all bits are programmed, in true scan lines)

10 bits in CR36[3:2] (msb-s) and CR38[7:0] (lsb-s).

- **SAdOf = Surrounding Address Offset** (a number to be added every line to the surrounding CRT Address Counter value to allow it to jump over the MVW without counting through it. This allows calculating the surrounding restart address.) This number depends on the surrounding mode resolution. Assuming a maximum line of 1024 pels, in increments of 4 pels per address (8bits/pel), 256 addresses is enough -> 8bits.

8bits in CR39[7:0]

- **WMAdS = MVW Memory Address Start** in increments of 16KB = 4KW (a physical address 00000:3FFFF for 1MB or 00000:7FFFF for 2MB Video Memory).

7bits in CR3A[6:0]

- **WMAdOf = MVW Memory Address Offset** (A number to be added to the current MVW Start Address to get the next MVW start Address). Having this s/w model allows panning through a large image for the MVW. The actual image stored in memory may be as large as 1K pixels

at 16bit per pixel (2 per word) -> 9bits.

9bits -> in CR36[4] (msb) and CR3B[7:0] (lsb).

Because the CRT Address Counter counts surrounding memory cycles, during the MVW display it would count a wrong number corresponding to the MVW pixel depth. To avoid this wrong count, the CRT Address Counter is stopped while MVW is displayed and loaded with the value corresponding to the end of MVW and restart of the surrounding display. As this address value changes every line, an offset is specified and will be programmed by the MS Windows driver depending on the MVW size and pixel format. The horizontal size of the window is controlled by a different counter counting to XW. During the MVW display the address will be generated by the same CRT Address Counter which will be loaded with a MVW Memory Address Start (WMAdS).

- **XW8P = MVW Horizontal Width in 8 pixel units.** Defines the size of the window in 8 pixel units. It is used to terminate the window horizontally. It may require to program the value minus one or two. To be Defined Later. Register CR3D[6:0] will contain this value.

Every scan-line

MVW Scan-Line Start Address = Previous Scan-Line MVW Start Address + WMAdOf

Scan-line doubling for "zoom" should be taken into account when designing the MVW Architecture: on the address generation block.

- **Off-screen MV Memory Addressing and Access Control.** This includes data type tag generation based on MV memory Address and the circuitry placing the tags in the CRT-FIFO. This block has provisions for scan line replication by repeating the MV address generated on the previous scan line of the MV window (only).

As Cinepak needs 4:1:1 rectangular format the MV Memory Address Counter needs to count in a non sequential way:

$Y0Sn, Y1Sn, Y2S(n+1), Y3S(n+1), U, V$

implies an address counter that can skip over two numbers, or maybe a second address counter that starts counting from 2 instead of zero.

- **One tag bit is generated based on CRT Address and is 1 if the data in the CRT FIFO is from the MVW, other-wise is 0.** This tag bit, called the **steering bit** will be delayed through the entire data path and end up controlling the final video data mux just before the DAC.

Other tag bits called "data-type" tag bits encode the type of 32bit word in a given data format and are used by MVW Decoder/Serialiser to steer CRT-FIFO data at CRT-FIFO read.

For instance, in 4:1:1 linear data format, data is stored in memory and in the CRT-FIFO as $Y0Y1Y2Y3, U03V03Y4Y5, Y6Y7, U47V47$, 3 groups of 32bits. Two "data-type tag bits will mark the three types of 32bit words so that steering logic knows where to place the three words in the data serialiser.

Similarly Sashapak will need 3 "data-type" tag bits to encode 8 32bit words (for 6bit per U/L) or 4 "data-type" tag bits to encode 10 32 bit words (for 8 bits per U/L).

16bit RGB 5:6:5 and 5:5:5 and 8bit RGB 3:3:2 can either use the standard VGA data

path, which in 5428 was extended to 16bit wide (used for 16bit/pel 1kx768 interlaced) or the new data path. If a physically new data path will be actually created.

- **MV Data Path**. This is area wise the largest piece. It consists of a new 16bit wide data-path with the same delay as the VGA data path, that takes MV data from the MV prefetch cache and the CRT FIFO, treats it appropriately depending on the MV data format encoded in the MV tags and converts it to 24 bit YUV first and 24bit RGB second. Horizontal zoom with averaging (or maybe 5 levels of interpolation) is in this block too.

- **Larger (26 or 24 stages) CRT-FIFO, Variable CRT-FIFO threshold and its control**. Because the MS Windows pixel depth and MVW pixel depth don't match, it is necessary to reserve CRT-FIFO space to fill in enough high pixel depth data before the MVW display starts. In Sashapak this means 10 or 8W (for 8 or 6bits per U/L respectively).

This function can be implemented as a dynamic threshold size modification of the CRT-FIFO. If the CRT-FIFO has 24 stages instead of 16 but only 16 stages are used before the MVW but it is switched to 24 when MVW fetches start, there will be 8 more stages to fill at that time ensuring more prefetched data in the CRT-FIFO at the beginning of each MVW scan-line.

In normal operation the threshold is 8 (or 6).

Here is the sequence of events that follow the detection of MVW start:

1. Drop CRT-FIFO threshold to 1 to get immediate service & make 10 (or 8) more stages available in the CRT-FIFO.
2. Start fetching MVW data till CRT-FIFO is full.
(BLT in progress will increase latency)
3. Increase threshold to 10.
4. Proceed as such till end MVW.
5. Load the CRT Address Counter with its last contents + SAdOf
(where SAdOf = Surrounding Address Offset)
7. Continue fetching based on the new address
(start with random cycle). CRT-FIFO threshold remains the same
- at least 10 or 8. CRT-FIFO size is still 26 or 24.
8. At the end of line, upon flushing the CRT-FIFO before prefetch, decrease CRT-FIFO size back to 16 keeping the threshold the same.
9. Go back to #1 for the next scan-line.

This sequence is applied if MVW pixel depth is higher than surrounding pixel depth. If MVW pixel depth is lower than surrounding pixel depth, then the action described at #1 above takes place at the end of MVW, instead of the beginning. *In other words, FIFO depth extension takes place when a switch from low to high pixel depth occurs.* If MVW pixel depth is the same as the surrounding, only the address is switched. In this case the CRT FIFO depth and the threshold remain the same.

- Steering logic decode the tags coming out of the special addition to the CRT-FIFO and the pre-fetch buffer and controls the decompression, formatting and serialization. (they tell the formatter what data is Y, U or V).

- What about the video serializer load and the CRT FIFO read signal? What happens to them when in MVW? They are one signal today.

WHEN the data path switched to display MVW data, what happens with the VGA Video Data Path is don't care as we do not display that data, except for the h/w cursor and the h/w icon that do not go through the standard VGA serializer. So we'll probably stop the VGA data path until the point h/w cursor and h/w icon are combined into it, to save power, as we will stop the MVW data path when not in use, for the same reason.

So, the Video Serializer Load may be stopped while MVW data is displayed, but the CRT FIFO read will be going on, with a frequency and shape depending on the data format loaded in the CRT FIFO for display.

In RGB 3:3:2 8bit/pel direct color mode CRT-FIFO read will be generated every 4 vclks. In standard RGB 5:5:5 or 5:6:5 16bit/pel modes CRT-FIFO read will be generated every two vclks.

In YUV 4:2:2 24bit/pel (Y0 U Y1 V on one scan line) the CRT-FIFO read will be generated also every two vclks.

In Sashapak 10 (for 8bit per U/L) or 8 (for 6bit per U/L) CRT-FIFO read pulses will be generated every 32 vclks.

- The MVW Address Counter, loaded at the beginning of each MVW scan-line with a start address calculated based on the Start MVW Address and the MVW Address Offset, will count at format dependent rate for as long as the CRT-FIFO is not full. Because the CRT-FIFO will be emptied faster in the MVW (in general the pixel depth will be higher in the MVW) there will be requests for more memory cycles when displaying in the MVW.

- If either h/w cursor or h/w icon are on, they should be displayed even if the direct data path is used. In this case both video data paths will clock and the steer tag will point to the VGA data path when either h/w cursor or h/w icon or both are displayed, even on top of a MV window. Please note that neither the h/w cursor, nor the h/w icon data go through the CRT FIFO or the VGA Data path except for the final pixel address block and the RAMDAC RAM, though they have a special path in the RAM. So we could stop most of VGA data path (including the internal palette) but not all of it even if we are in 16bit/pel mode (the new one that goes through a separate 16bit data path).

- If we are in a 16bit/pel mode with no h/w cursor or h/w icon, or in a 640x480 MVW mode filling the screen, we can stop the VGA data path to save power.

- The off-screen MVW memory will be a programmable start memory array normally placed towards the end of the memory, but before h/w cursor, h/w icon and the half frame buffer, even the color one. The start of the MVW memory will be programmable in increments of 4KW (16KB) as a physical address -> 7bits to program.

CR3B[6:0] is the MVW Memory Array Start Address.

2MB configuration will support enough memory to support:

- one 640x480 3bit/pel window or up to two 320x240 3bit/pel (Sashapak) -> 32KW
- one 640x480 16bit/pel window or two 320x240 16bit/pel windows (153.6KW=614.4kB out of 256KW or 512KW available depending on the memory configuration).

- all of the above at the same time up to a total of two windows though.

1MB configuration will support:

- one 640x480 3bit/pel Sashapak window or two 320x240 Sashapak windows or
- two 320x240 16bit/pel windows.

Please note that about 24KW are needed for 800x600 DS Color STN HFB, the h/w cursors and h/w icons while about 18KW are needed for 640x480 DS Color STN HFB and that

- 640x480 4bit/pel requires 38.4KW
- 800x600 4bit/pel requires 60KW
- 1024x768 4bit/pel requires 98.3KW
- 640x480 8bit/pel requires 76.8KW
- 800x600 8bit/pel requires 120KW
- 1024x768 8bit/pel requires 196.608KW

Video Motion Formats supported are:

- 8bit RGB 3:3:2 going through the palette (RGB 3:3:2)
- 16bit RGB (5:5:5 Sierra and 5:6:5 XGA) not going through the palette -> this is a 16bit/pel that does not require double dotclock !!
- 4:1:1 array converted to 4:2:2 Y0 Y1 Y2 Y3 U V -> Y0UY1V Y2UY3V for Cinepak
(with Y2Y3 for the next scan line)
- 4:2:2 linear Y0UY1V from TV Decoder
- Sashapak format (we'll support only 6bit/pel)
 - for 8bit per U/L:
 $4W(YU/L) + 4W(UV\ U/L) + 1W(YM) + 1W(UVM) = 10W = 40B$
--> 40B/32pixels = 320bit/32pixels = 10bit/pel to fetch from memory
 - for 6bit per U/L:
 $3W(YU/L) + 3W(UV\ U/L) + 1W(YM) + 1W(UVM) = 8W = 32B$
--> 32B/32pixels = 8bit/pixel

CR3C[3:0] will encode the pixel format for the first motion video window data:

- 0h = 0000 = 8bit RGB 3:3:2 going through the palette
- 1h = 0001 = 16bit RGB Sierra (5:5:5)
- 2h = 0010 = 16bit RGB XGA (5:6:5)
- 3h = 0011 = 4:2:2 YUV Y0UY1V
- 4h = 0100 = 4:1:1 YUV linear Y0Y1Y2Y3UV
- 5h = 0101 = 4:1:1 YUV array Y0Y1Y2Y3UV but Y2Y3 are in next scan line
- 6h = 0101 = Sashapak 6bits for U/L
- 7h = 0110 = Sashapak 8bits for U/L
- 8h:Fh = 0111:1111 = reserved

- CR3C[4] = Enable Motion Video Window
- CR3C[5] = Enable Live Video in Full Screen
- CR3C[7] = MVW horizontal zoom (doubling)
- CR3C[6] = MVW vertical zoom (doubling)

**Cirrus Confidential
Business Information**

CL 99801

Note: We will not support 4:1:1 linear from TV decoder, for design simplicity.

Memory data formats:

- 8bit and 16bit RGB, as in 5428.
- 4:1:1 Cinepak converted to 4:2:2 by writing a 4:2:2 Code-Book
first scan line -> Y0UY1V -> One 32bit Word

second scan line in memory -> Y2UY3V -> One 32bit Word
- 4:2:2 linear will be mapped in memory as follows:

first scan line Y0UY1V
first scan line Y2UY3V

consecutive data in the same scan line.

The tags that accompany the motion video data through the CRT-FIFO mark both the data format and the type of data inside the format (U/L for Y or U,V or mask for Y or U,V in Sashapak, Y0, Y1,U or V in 4:2:2).

Double Buffering for MVW Video Memory Data

In order to get a high quality Live Video full frames should be displayed at all times. This assumes that writing the video buffer and displaying it are in sync, which is normally not the case.

To solve this problem and provide a simple interface between MVA h/w and the driver, Nordic will support a double buffer approach.

Addressing with Sashapak:

Sashapak stores data for eight scan-lines in one memory page (works only with synchronous DRAM-s). But due to U,V sub-sampling different data types are not read from memory uniformly.

On Y U/L, the first four scan-lines read at offset 0 in the page, but the second four scan-lines read at offset 50H in the page.

On U V U/L, we read the same data for each scan-line. So we repeat reading the same data at all times, for eight scan-lines. Page addresses to read are A0:EF. If the scan-line is shorter it will end sooner.

On Y Map the data should be sequential, scan-line oriented: first pixels 0 to 639 on the first scan line, then pixels 0 to 639 on the second scan line and so on till the 8-th scan line. If the lines have less than 640 pixels, Sasha wants them cropped for the first 4 scan lines, and then starting again from the half for the next four scan-lines. This is in sync with Y U/L where the last four scan-lines start from mid.

On U V Map, because of the vertical subsampling, the same data is read for two scan lines, so there are four scan-lines worth of data but each scan-line is read for two consecutive scan-lines. Even if the scan-line is shorter, there will be a gap every-scan-line: there are fix page addresses for each scan-line up to 640 pixels wide.

Cirrus Confidential
Business Information

CL 99802

Memory data formats:

- 8bit and 16bit RGB, as in 5428.

- 4:1:1 Cinepak converted to 4:2:2 by writing a 4:2:2 Code-Book
first scan line -> Y0UY1V -> One 32bit Word

second scan line in memory -> Y2UY3V -> One 32bit Word

- 4:2:2-linear will be mapped in memory as follows:

first scan line Y0UY1V

first scan line Y2UY3V

consecutive data in the same scan line.

The tags that accompany the motion video data through the CRT-FIFO mark both the data format and the type of data inside the format (U/L for Y or U,V or mask for Y or U,V in Sashapak, Y0, Y1,U or V in 4:2:2).

Double Buffering for MVW Video Memory Data

In order to get a high quality Live Video full frames should be displayed at all times. This assumes that writing the video buffer and displaying it are in sync, which is normally not the case.

To solve this problem and provide a simple interface between MVA h/w and the driver, Nordic will support a double buffer approach.

Addressing with Sashapak:

Sashapak stores data for eight scan-lines in one memory page (works only with synchronous DRAM-s). But due to U,V sub-sampling different data types are not read from memory uniformly.

On Y U/L the first four scan-lines read at offset 0 in the page, but the second four scan-lines read at offset 50H in the page.

On U V U/L we read the same data for each scan-line. So we repeat reading the same data at all times, for eight scan-lines. Page addresses to read are A0:EF. If the scan-line is shorter it will end sooner.

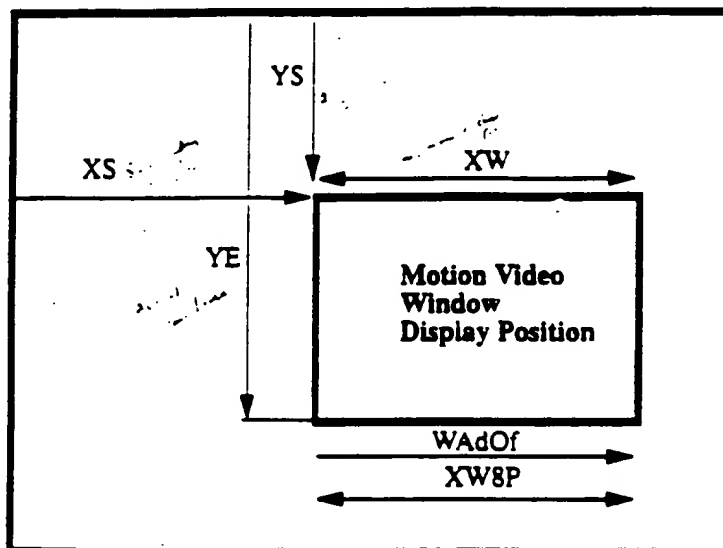
On Y Map the data should be sequential, scan-line oriented: first pixels 0 to 639 on the first scan line, then pixels 0 to 639 on the second scan line and so on till the 8-th scan line. If the lines have less than 640 pixels, Sasha wants them cropped for the first 4 scan lines, and then starting again from the half for the next four scan-lines. This is in sync with Y U/L where the last four scan-lines start from mid.

On U V Map, because of the vertical subsampling, the same data is read for two scan lines, so there are four scan-lines worth of data but each scan-line is read for two consecutive scan-lines. Even if the scan-line is shorter, there will be a gap every-scan-line: there are fix page addresses for each scan-line up to 640 pixels wide.

Cirrus Confidential
Business Information

CL 99803

Full Screen in a given graphics mode



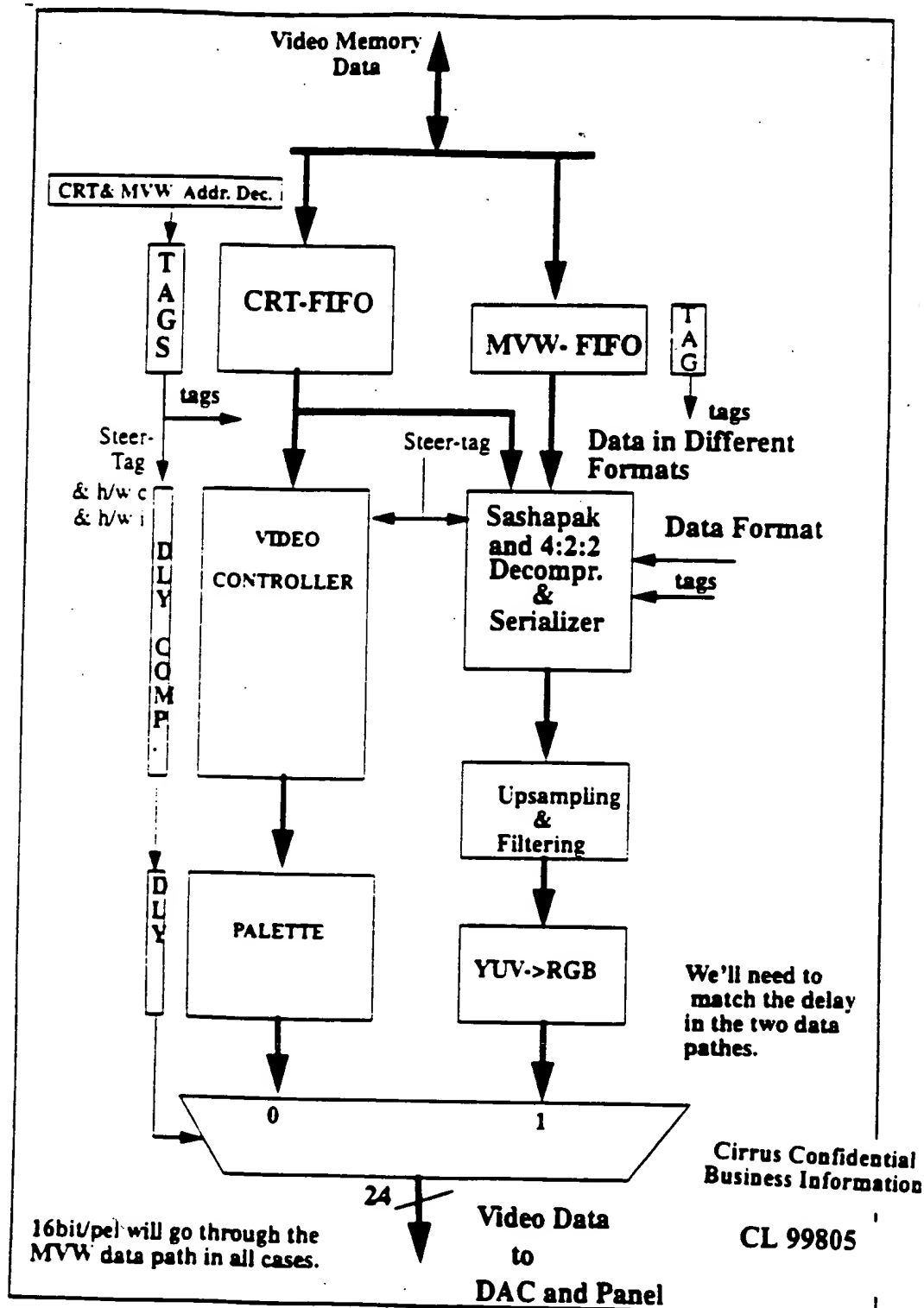
Where:

- XS = MVW Horizontal Start Coordinate (in surrounding pel depth memory cycles)
- XW = MVW Horizontal Width (MVW pel depth memory cycles)
- YS = Vertical MVW Start (in true scan-lines not affected by scan-line doubling or panel vertical expansion)
- YE = Vertical MVW End
(all bits are programmed, in true scan lines)
- WAdOf = MVW Address Offset (a number to be added every line to the CRT Address Counter to allow it to jump over the MVW without counting through it.
- XW8P = MVW horizontal width expressed in 8 pixel units

Please note that MVW horizontal width is defined twice in different units:
 XW defines it on the memory side in memory cycles
 XW8P defines it on the CRT side in 8 pixel units.

**Cirrus Confidential
 Business Information**

CL 99804



5.0 Registers for Motion Video Architecture**5.1 CR34 - MVW XS Register**

Bit	Definition
[7:0]	MVW Horizontal Start Coordinate XS (in surrounding pel depth memory cycles) The horizontal position is programmed with 8 pixel resolution (0,8,16,... pixels starting from the left side of the screen) (up to 1K pels in increments of 8 pels). To get an early warning, we will require XS to be programmed 8 pixels less than the actual position: 0 = zero, 1 = 16 pels, 2 = 24 pels, etc.

5.2 CR35 - MVW Horizontal Width XW Register

Bit	Definition
[7:0]	MVW Horizontal Width XW (MVW pel depth memory cycles) (up to 1K pels in increments of 8 pels). To get an early warning, we will require XW to be programmed 8 pixels less than the actual width: 0 = zero, 1 = 16 pels, 2 = 24 pels, etc. width.

5.3 CR36 - Vertical MVW High Position Register

Bit	Definition
[7]	Reserved
[6]	Cinepak type YUV to RGB conversion. If this bit = 0, YUV to RGB conversion is done according to the standard algorithm. If this bit = 1, the YUV to RGB conversion is done based on Cinepak algrtm.
[5]	Excess 128 for YUV to RGB conversion. U and V are integer values expressed as two-th complement or excess 128. If this bit = 0, the YUV to RGB converter assumes that U and V are expressed in 2-th complement format (as in Philips SAA715B and SAA9051 TV Decoders). If this bit = 1, the YUV to RGB converter assumes that U and V are expressed in excess 128 format.
[4]	MVW Memory Address Offset (WMAdOf MSB)
[3:2]	Vertical MVW End MSBs (see CR37)
[1:0]	Vertical MVW Start MSBs (see CR38)

Cirrus Confidential
Business Information

CL 99806

5.4 CR37 - Vertical MVW Start Register

Bit	Definition
[7:0]	Vertical MVW Start YS: (in true scan-lines not affected by scan-line doubling or panel vertical expansion) 10 bits in CR36[1:0] (msbs) and CR37[7:0] (lsbs).

**Cirrus Confidential
Business Information****CL 99807**

5.5 CR38 - Vertical MVW End YE Register

Bit	Definition
[7:0]	Vertical MVW End YE: (all bits are programmed, in true scan lines) 10 bits in CR36[3:2] (msb-s) and CR38[7:0] (lsb-s).

5.6 CR39 - Surrounding Address Offset Register

Bit	Definition
[7:0]	Surrounding Address Offset: SAdOf (a number to be added every line to the surrounding CRT Address Counter value to allow it to jump over the MVW without counting through it. This allows calculating the surrounding restart address.) This number depends on the surrounding mode resolution. Assuming a maximum line of 1024 pels, in increments of 4 pels per address (8-bits/pel), 256 addresses is enough for eight bits

5.7 CR3A - MVW Memory Address Start Register

Bit	Definition
[6:0]	MVW Memory Address Start: WMAdS in increments of 16KB = 4KW (a physical address 00000:3FFFF for 1MB or 00000:7FFFF for 2MB Video Memory). Note that on the CPU side, this Start Address should be aligned at 16KB.

5.8 CR3B - MVW Memory Address Offset Register

Bit	Definition
[7:0]	MVW Memory Address Offset WMAdOf (A number to be added to the current MVW Start Address to get the next MVW start Address). Having this s/w model allows panning through a large image for the MVW. The actual image stored in memory may be as large as 2K pixels at 16 bits per pixel (two per word) -> 10bits. 10bits -> in CR36[5:4] (msb) and CR3B[7:0] (lsb).

**Cirrus Confidential
Business Information**

CL 99808

5.9 CR3C - MVW Pixel Format Register

Bit	Definition
[7]	MVW horizontal zoom (doubling)
[6]	MVW vertical zoom (doubling)
[5]	Enable Live Video in Full Screen
[4]	Enable Motion Video Window
[3:0]	<p>Encode format for the first Motion Video Window as well as full screen graphics going through the MVA data path at 1x dot-clock frequency:</p> <p>0h = 0000 = 8 bit RGB 3:3:2 direct (not through the palette), going through MVA</p> <p>1h = 0001 = 16 bit RGB Sierra (5:5:5) (at 1x dot-clock frequency)</p> <p>2h = 0010 = 16 bit RGB XGA (5:6:5) (at 1x dot-clock frequency)</p> <p>3h = 0011 = 4:2:2 YUV Y0UY1V</p> <p>4h = 0100 = Reserved for 4:1:1 YUV linear Y0Y1Y2Y3UV</p> <p>5h = 0101 = Reserved for 4:1:1 YUV array Y0Y1Y2Y3UV but Y2Y3 are in next scan line</p> <p>6h = 0110 = Sashapak 6 bits for U/L</p> <p>7h = 0111 = Sashapak 8 bits for U/L</p> <p>8h = 1000 = 24bit RGB through MVA (at 1x dot-clock frequency)</p> <p>9h = 1001 = 8 bit RGB through the palette</p> <p>For improved quality of 8 bit/pel. A program can even update the palette during 8bit live video to get more colors.</p> <p>ah = 1100 = 8 bit direct color grey shaded (R=G=B=8bit grey shade)</p> <p>To be used for image processing in the printing industry related applications. This mode Nordic can run at 1024x768 as any other 8bit/pel.</p> <p>Bh:Fh = 1001:1111 = reserved for future formats</p> <p>When a Motion Video Window is created data always goes through the MVA data path. In this case only the VGA Data Path can be used for the surrounding data.</p> <p>If no Motion Video Window, the MVA data path can be used to display full screen in any of the above formats. Especially for high pixel depth modes (24, 16bpp) it is recommended to use the MVA data path which runs at 1X dot-clock frequency and has a larger FIFO than the VGA Data Path.</p> <p>A full screen motion video window can be also defined that can be anywhere in memory and can run video or any application in the above formats.</p>

5.10**CR3D - Motion Video Window Horizontal Size Register**Cirrus Confidential
Business Information

Bit	Definition
[7]	<p>MVA Data Bus Signature Generation Selection bit.</p> <p>0 = selects VGA Data Path for signature generation</p> <p>1 = selects MVA Data Path for signature generation</p>
[6:0]	<p>Motion Video Window width in 8 pixel units</p> <p>This register is used to specify the horizontal width of the MVW in 8 dot-clock</p>

units.

The MVW has to be a multiple of 8 pixels. The value programmed in this register is used to terminate the MVW on each scan line.

**Cirrus Confidential
Business Information**

CL 99810

6.0 Registers for Panel Horizontal Timing Control

This set of four registers are used to generate horizontal panel timing with 640x480 or 800x600 panels and to center horizontally a 640dots or 720dots picture on an 800x600 panel.

6.1 CR40 - No Center Panel HDE Start Register

Bit	Definition
[7:0]	Panel Horizontal Display Enable Start relative to previous HDE start, in 4 dot-clock units. The dot-clock used is never divided by two. This register is used to program Panel Line Clock Start and Panel Display Enable Start with both 640x400 and 800x600 panels any time no horizontal centering is needed. Used automatically by h/w if CR41 and CR42 are not used.

6.2 CR41 - Panel HDE Start Register to Center 720 dots

Bit	Definition
[7:0]	Panel Horizontal Display Enable Start relative to previous HDE start, in 4 dot-clock units. The dot-clock used is never divided by two. The value in this register will be automatically used by Nordic to control Panel Line Clock and Panel Horizontal Display Enable Start anytime a 720 dot mode (text or graphics - RIX mode only) is centered on an 800x600 panel. If horizontal expansion is on, CR40 is used. Used automatically by h/w if : 800x600 panel & No 800x600 graphics mode & 8 dot character clock & no horizontal expansion

6.3 CR42 - Panel HDE Start Register to Center 640 dots

Bit	Definition
[7:0]	Panel Horizontal Display Enable Start relative to previous HDE start, in 4 dot-clock units. The dot-clock used is never divided by two. The value in this register will be automatically used by Nordic to control Panel Line Clock and Panel Horizontal Display Enable Start anytime a 640 dot mode (text or graphics) is centered on an 800x600 panel. If horizontal expansion is on, CR40 is used. Used automatically by h/w if : 800x600 panel & No 800x600 graphics mode & (9 dot character clock + horizontal display enable $\geq 50H$) & no expansion no horizontal expansion

Cirrus Confidential
Business Information

CL 99811

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ **BLACK BORDERS**
- ☒ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☒ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☒ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.